computational considerations

and apparently awesome alliterations

-

16. joulukuuta 2020

what about that quantum

"Quantum mechanics is a trivial theory" -Unnamed Instructor

$$\hat{H}_{JC} = \frac{\hbar\Omega}{2}(\hat{a}\hat{\sigma}_+ + \hat{a}^\dagger\hat{\sigma}_-) + \hbar\omega_c\hat{a}^\dagger\hat{a} + \hbar\omega_a\frac{\hat{\sigma}_z}{2} \tag{1}$$

$$\hat{\sigma}_+ = |g\rangle\langle e| \tag{2}$$

$$\hat{\sigma}_- = |e\rangle\langle g| \tag{3}$$

stuff in hilbert space = matrices

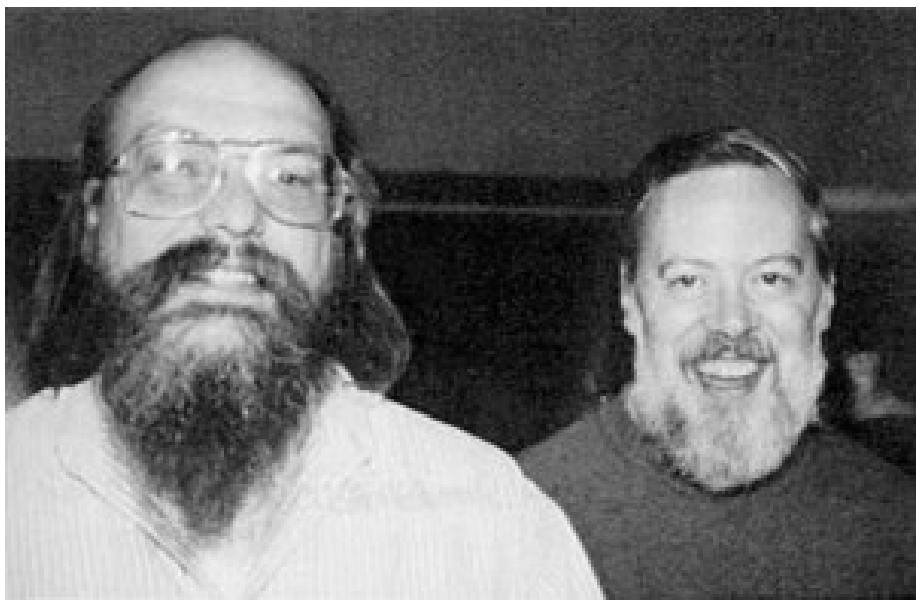
tensor product = matrices

everything is matrices

matrices: a solved problem

nice things about matrices

- There are numerous libraries available for matrix operations (didn't use them tho)
- Written by wizards in the 70s (see picture)
- Run faster than me confronted with the prospect of having to socialize



Kuva: Average numerical linear algebra alpha male

matrices: a solved problem

infinite matrices??

$$\hat{a} = \frac{1}{\sqrt{2}}(\hat{x} + i\hat{p}) \tag{4}$$

$$\hat{a}^\dagger = \frac{1}{\sqrt{2}}(\hat{x} - i\hat{p}) \tag{5}$$

matrix rep :

$$\hat{a} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & \sqrt{2} & 0 & \dots \\ 0 & 0 & 0 & \sqrt{3} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{6}$$

matrices: a solved problem

truncation and error

$$\hat{a} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & \sqrt{2} & 0 & \dots \\ 0 & 0 & 0 & \sqrt{3} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \hat{a}^\dagger = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & \dots \\ 0 & \sqrt{2} & 0 & 0 & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots \end{bmatrix}$$

need to impose a cutoff at some n, but how does error decrease?

ERROR = % OF ELEMENTS THAT DON'T MATCH

matrices: a solved problem

truncation and error

$$[\hat{a}, \hat{a}^\dagger] = 1 \tag{7}$$

Matrix rep, $n = 5$ and $n = 6$

$$[\hat{a}, \hat{a}^\dagger] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, [\hat{a}, \hat{a}^\dagger] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{error} = 20\% \text{ and } = 18\%$$

matrices: a solved problem

truncation and error

so: error decreases as O(n) but matrix size increases as $O(n^2)$.

```
for(int i = 0; i < rows; i++) {
        for(int j = 0; j < columns, j++) {
                ...
```

so execution time: $O(n^2)$, sometimes worse. not ideal, can't be helped!

Why are QM problems computationally difficult?

Computer memory

computer is a stupid rock tricked in to thinking!

memory is aligned, meaning:

- processor starts reading RAM at addresses divisible by 64(8)
- 4-byte variables must be in a memory slot divisible by 4, 6 byte variables in spots divisible by 6 etc
- reading RAM is S L O W

Why are QM problems computationally difficult?

Computer memory

illustration:

```
struct Person {
        char first_letter_of_name; // 1 byte
        int age;                   // 4 bytes
        short number_of_kids;      // 2 bytes
}
```

```
struct Person {
        char first_letter_of_name; // 1 byte
        char padding[3]            // 3 bytes
        int age;                   // 4 bytes
        short number_of_kids;      // 2 bytes
}
```

in code: 7 bytes = single read by CPU

in memory: 10 bytes = two reads by CPU = 2x slower!

it takes 2-10 ns to access CPU cache, 200+ns to access RAM

fix: "memory packing" -> reorder the elements according to alignment. NOT POSSIBLE in anything except low level languages

QM = huge amounts of data, thus huge amounts of accesses, thus a difficult computational problem

Example 1

Semiclassical Rabi

$$\dot{C}_l(t) = -\frac{i}{\hbar} \sum_k C_k(t) \langle l| H' |k\rangle \tag{8}$$

in the code (under the rotating wave approximation):

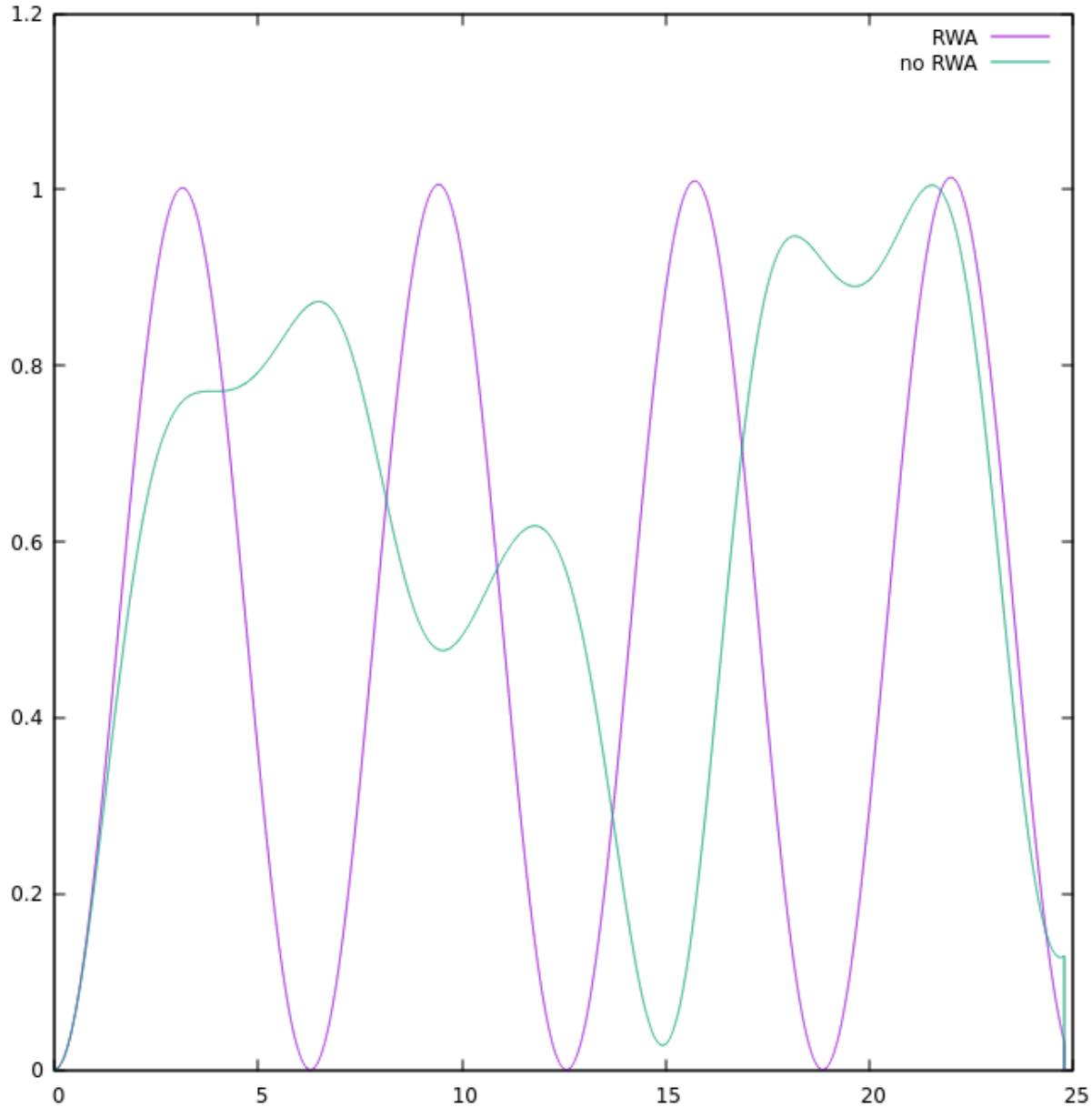
```
complex double C1 = I/2 *matmul(p,b0,matmul(p,d,k1))
->matrix[0][0]*x2*cexp(I*detuning*t);
complex double C2 = I/2 *matmul(p,b1,matmul(p,d,k0))
->matrix[0][0]*x*cexp(-I*detuning*t);
```

also, python code to interactively run the simulation:

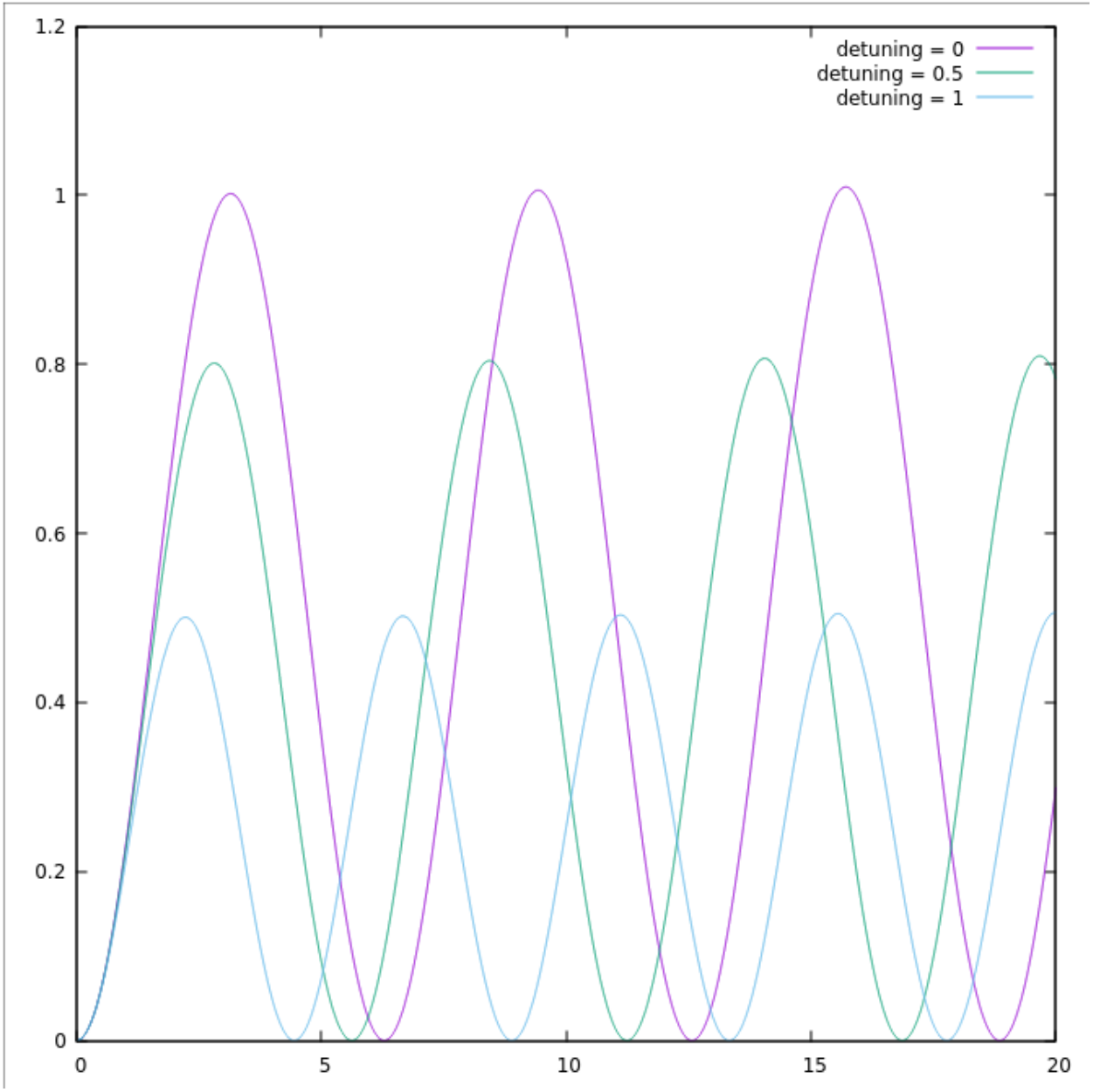
```
import scqo
system = scqo.NLevelAtom(omega = 0.5)
field = scqo.ClassicalField(omega = 0.5)
system.calculate()
#no rwa
system.set_rwa(False)
system.calculate()
```

Plot the results:

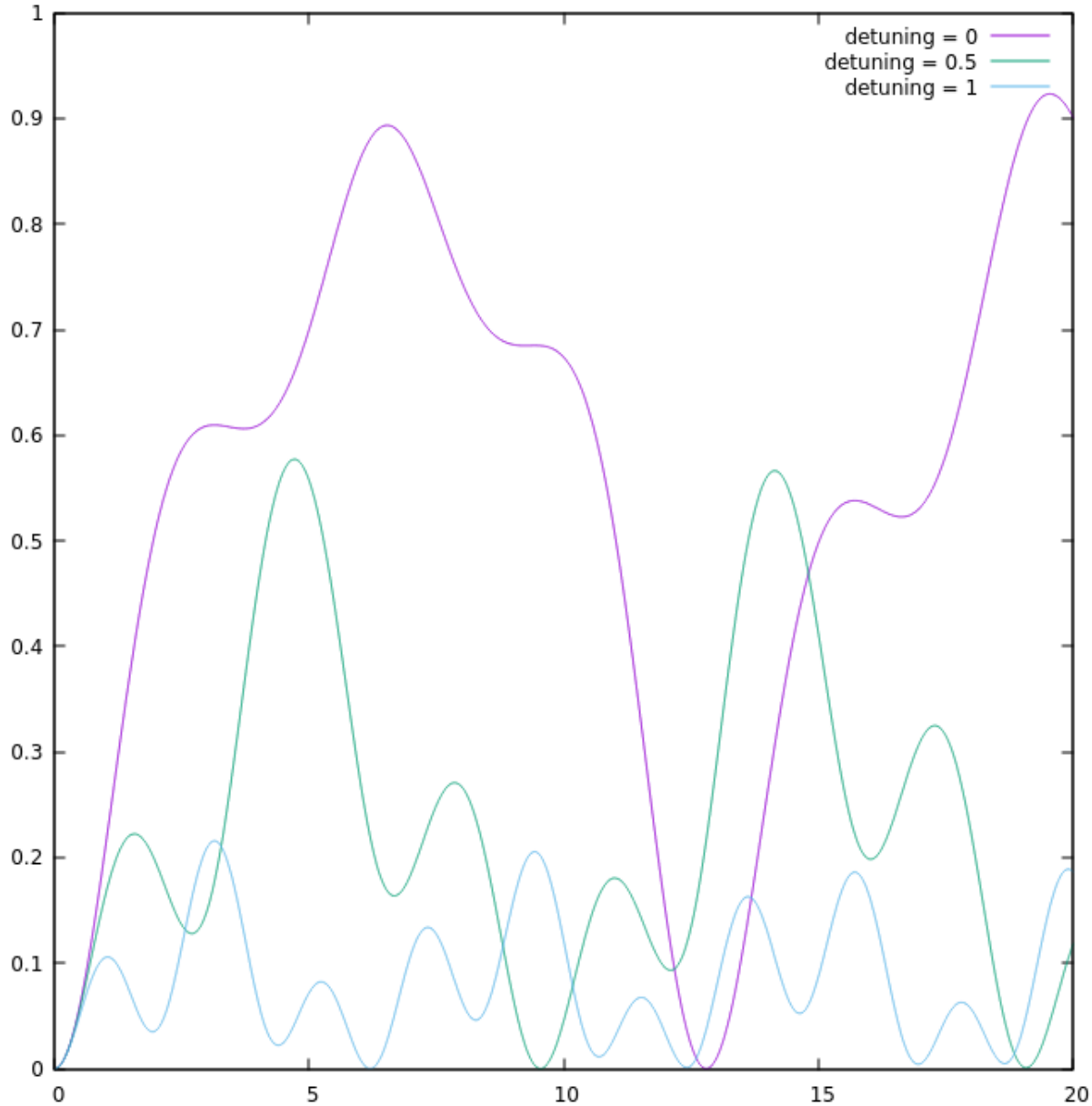
lesson about approximations: they might actually suck. cows are not always spherical



Different detunings.



..with no RWA


detuning = 0
detuning = 0.5
detuning = 1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20

Example 2

Bose-Hubbard dimer (=two sites)

Bose-Hubbard timer: lattice sites + particles interacting.

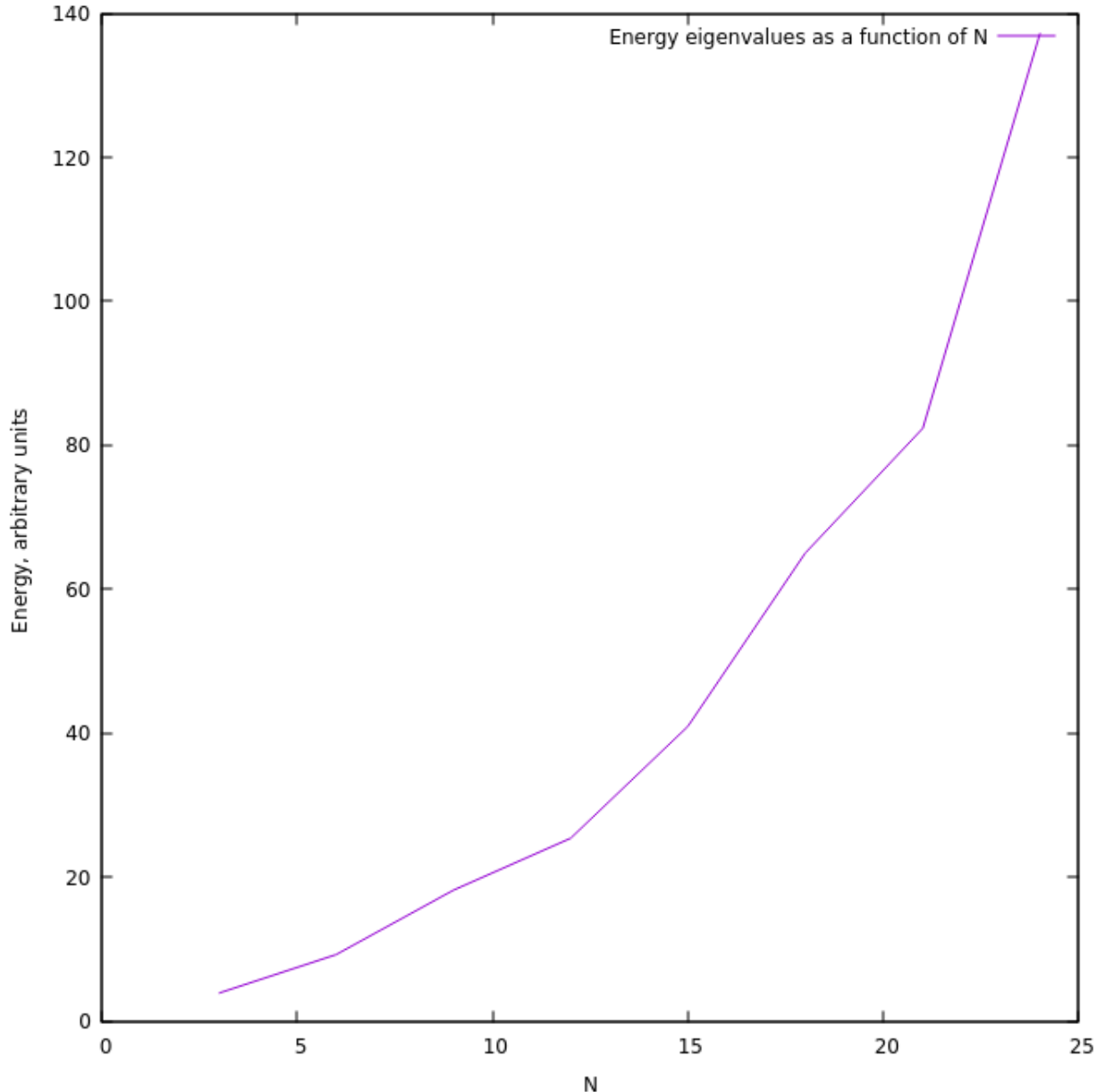
$$\hat{H} = \epsilon(\hat{a}_1^\dagger \hat{a}_1 - \hat{a}_2^\dagger \hat{a}_2) + v(\hat{a}_1^\dagger \hat{a}_2 + \hat{a}_2^\dagger \hat{a}_1) + c(\hat{a}_1^\dagger \hat{a}_1 - \hat{a}_2^\dagger \hat{a}_2)^2 \qquad (9)$$

huge matrix size for larger particle numbers, takes forever to run.

easy to implement with matrix methods

Example 2

Bose-Hubbard dimer (=two sites)



computational note: you can remove 0-padding from Hamiltonians without losing any important information about the energy levels:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & c & d & 0 \\ 0 & f & b & c & d & 0 \\ 0 & a & p & q & d & 0 \\ 0 & a & b & c & d & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \iff H = \begin{bmatrix} a & b & c & d \\ f & b & c & d \\ a & p & q & d \\ a & b & c & d \end{bmatrix} \tag{10}$$

Only zeroed eigenvalues are lost, and there are $2n$ of them, where n is the amount of padding removed.

Example 3: Simulating dynamics

Bose-Hubbard Linblad simulation

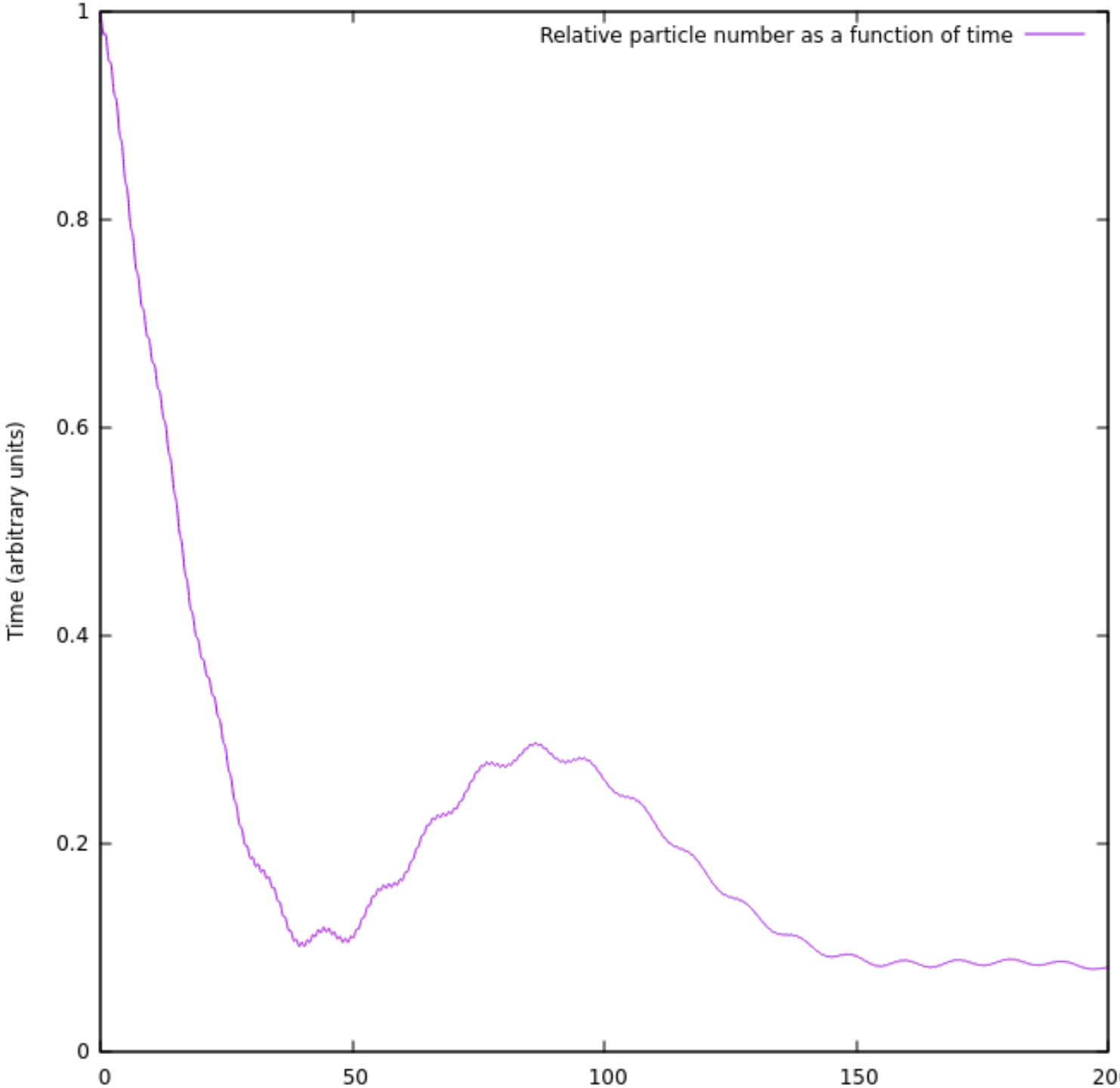
The Bose-Hubbard model as an open system (Lindblad form):

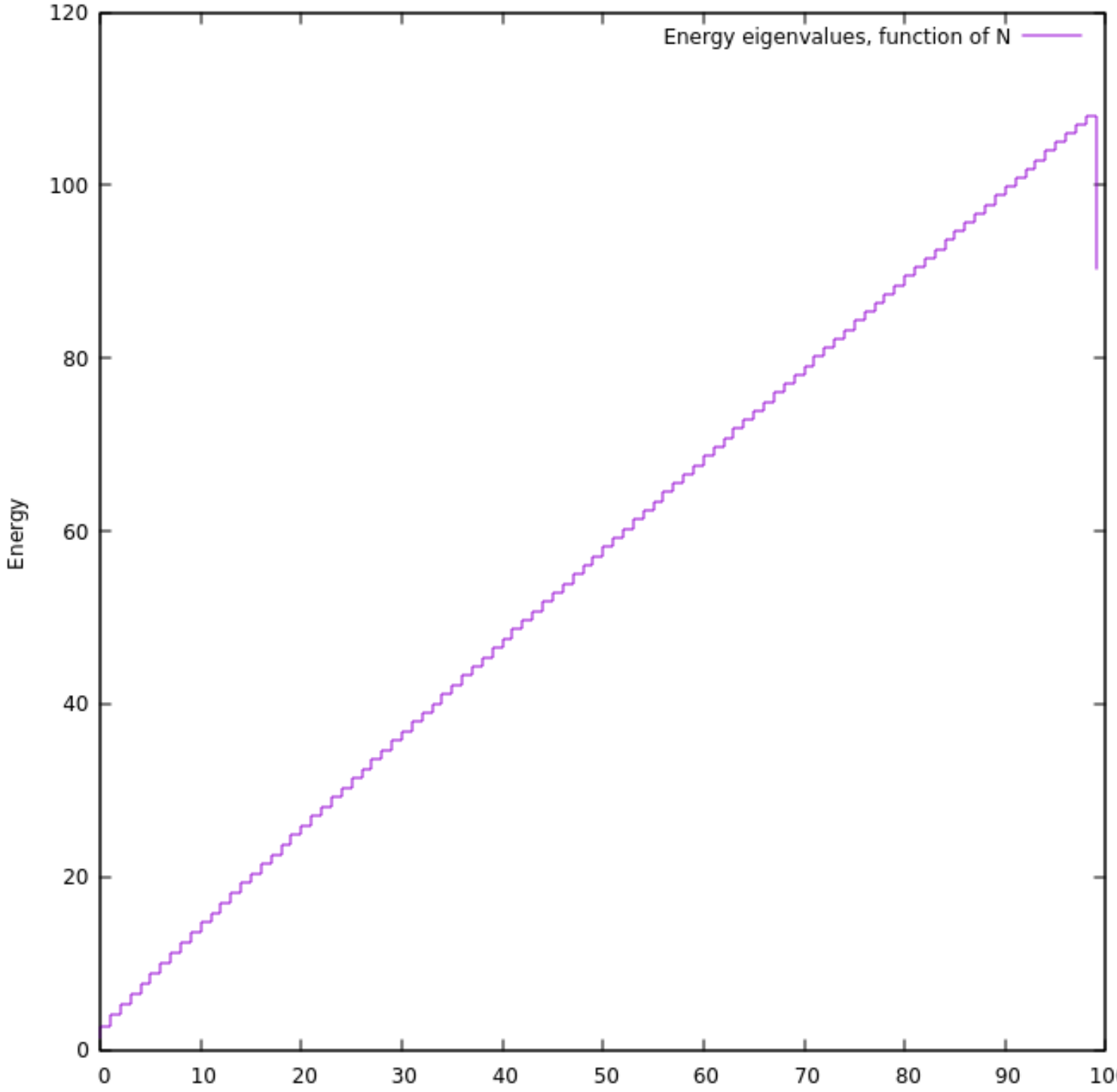
$$\dot{\hat{\rho}} = -i[\hat{H}, \hat{\rho}] - \frac{\gamma}{2}\left(\hat{a}_2^\dagger \hat{a}_2 \hat{\rho} + \hat{\rho}\hat{a}_2^\dagger \hat{a}_2 - 2\hat{a}_2 \hat{\rho} \hat{a}_2^\dagger\right) \quad (11)$$

$$\hat{H} = \epsilon(\hat{a}_1^\dagger \hat{a}_1 - \hat{a}_2^\dagger \hat{a}_2) + v(\hat{a}_1^\dagger \hat{a}_2 + \hat{a}_2^\dagger \hat{a}_1) + c(\hat{a}_1^\dagger \hat{a}_1 - \hat{a}_2^\dagger \hat{a}_2)^2 \quad (12)$$

H is the Hamiltonian discussed previously. How about dem dynamics?

(optical tunneling from the dimer to optical lattice)


Relative particle number as a function of time
Time (arbitrary units)
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200


Energy eigenvalues, function of N
Energy
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70
80
90
100

Some conclusions

- Matrices are a straightforward way of doing OQS
- The method is a bit slow (could be heavily optimized)
- Special care is needed to choose the right algorithms for unstable problems
- Approximations sometimes suck
- I hate segfaults